System of Automated Deduction (SAD): Linguistic and Deductive Peculiarities

Alexander Lyaletski[1], Konstantine Verchinine[2], Anatoli Degtyarev[3], and Andrey Paskevich[1]

[1] Cybernetics Department, Taras Shevchenko Kyiv National University, Ukraine
tel.: +38 (044) 266 3108, e-mails: lav@tc.unicyb.kiev.ua, andrey@raptor.kiev.ua

[2] Math-Info Department, Paris 12 University, France
tel.: +33 (6) 851 291 2022, e-mail: verko@logique.jussieu.fr

[3] Department of Computer Science, University of Liverpool, United Kingdom
tel.: +44 151 794 6792, e-mail: a.degtyarev@cs.man.ac.uk

Abstract. In this paper a state of the art of a system of automated deduction called SAD is described *. An architecture of SAD corresponds well to a modern vision of the Evidence Algorithm programme, initiated by Academician V.Glushkov. The system is intended for accumulating mathematical knowledge and using it in a regular and efficient manner for processing a self-contained mathematical text in order to prove a given statement that always is considered as a part of the text. Two peculiarities are inherent in SAD: (a) mathematical texts under consideration are formalized with the help of a specific formal language, which is close to a natural language used in mathematical publications; (b) proof search is based on a specific sequent-type calculus which gives a possibility to formalize "natural reasoning style". The language may be used as a tool to write and verify mathematical papers, theorems, and formal specifications, to perform model checking, and so on. The calculus is oriented to constructing some natural proof search techniques such as definition and auxiliary proposition applications.

Keywords: Automated Theorem Proving, Calculus, Completeness, Deduction, First Order Logic, Formal Language, Sequent, Validity

* Partially supported by INTAS 2000-447.

1 Introduction

In this paper some linguistic and deductive peculiarities of System of Automated Deduction (SAD) constructing now are described. These peculiarities satisfy well to the main principles of the realization of the Evidence Algorithm, or EA.

The Evidence Algorithm was advanced by Academician V.Glushkov as a programme of investigations in automated theorem proving. Its main objective was to help to working mathematicians in mathematical text processing, i.e. in computer-aided constructing and verifying long, but in some sense "evident" proofs. That is why V. Glushkov proposed to explore simultaneously: formalized languages for presenting mathematical texts in the form most appropriate for a user, a formal notion of evolutionary developing computer-made proof step, EA information environment that has influence on the evidence of a proof step, and man-assisted search for a proof.

An idea underlying SAD corresponds well to the contemporary trends of computer mathematical services construction. In this connection, we must pay your attention to the following.

A specific feature of SAD is that proof search of a theorem T under consideration is done in a framework of a self-contained mathematical text Txt written in a formal language close to a natural language used in mathematical papers. The term "Txt is a self-contained text (w.r.t. T)" denotes that Txt contains all the necessary for proving T. So, such a language should have a formalized syntax and semantics. It should permit the formulation of theory axioms, lemmas, theorems and proofs along with definitions in order to provide self-contained texts. Accordingly, the thesaurus of the language should be separated from its grammar in order to be extendible. Besides, the language should be close to the natural ones used in mathematical papers (this provides a user-friendly interface for creating a text or processing it in an interactive mode).

According to EA, the core of a mathematical text processing technique is a so-called "evidence routine" that establishes the evidence of a proven (verified) step in terms of some deductive formalism, which should permit: to preserve a structure of an initial problem; to search for an inference in the signature of an initial theory; to reduce a "goal" under consideration to a number of new auxiliary "subgoals"; to separate a deductive process from finding solutions for "equations" (for systems of "equations"); to use a specific equality handling technique; to build-in human-like methods of theorem proving such that, for example, definition and auxiliary proposition applications; to organize a flexible interactive search mode.

To achieve the above, various tools enforcing the "evidence routine" should be provided: search for auxiliary information, use of analytical transformations, application of a proof technique usual for a man, and so on. It is clear that first of all, both the tools and the "evidence routine" should be able to exchange data using a certain formalized language (languages). That is why this paper reflects a current state of investigations relating to the questions of linguistic and deductive support of SAD in accordance with the EA programme.

2 Linguistic Tools

A number of languages has been constructed for EA (see, for example, [1, 2]). By now, ForTheL (FORmal THEory Language) [3] is the last representative of this languages family. The main objective of the construction of ForTheL is to provide an initial environment for the evolutionary development of the "evidence routine" and to enforce proving tools of SAD depending on a problem to be solved and a mathematical text to be processed. Thus, ForTheL may be used as a tool to write and verify mathematical publications and formal specifications. Also, it may be used as a universal interface of declarative mathematical knowledge bases and as a tool for the integration of computer mathematical systems.

Basically, a ForTheL-text can be viewed as a collection of phrases possessed of some structure that provides certain "navigation rules" in the text. Therefore, we can distinguish two components of the language, which are, in some sense, orthogonal to each other. The first one concerns ForTheL-phrases only, and the second one defines the composition rules of a whole ForTheL- text – phrase grouping, section naming, etc.

The component concerning ForTheL-phrases is an extension of the first-order language by certain natural expressions. This allows to write ForTheL-phrases as phrases of a human- like language. In particular, the key features of the language are so-called concepts that represent classes of objects and are written as nouns.

Phrases in the ForTheL-text are grouped in sections such as axioms, theorems, definitions and so on. Note, that the language allows to define new constructions (concepts, predicates, functions)directly in a ForTheL-text under consideration.

According to a current approach, the processing of a mathematical text in the EA-style is made by the following scheme.

First of all, a text to be processed, should be written as a ForTheL-text. Then, the ForTheL-text must be automatically translated into a so-called ForTheL1-text. Any ForTheL1-text consists of sentences that, on the one hand, are analogs of 1st-order logic formulas, and, on the other hand, preserve a signature of an original ForTheL-text, its syntax and structure, i.e. partitioning into sections such as definitions, auxiliary propositions, and a theorem to be proved. After transforming a ForTheL-text into its ForTheL1-representation, the construction of a ForTheL1-environment for proof search procedures must be made.

In the frame of this ForTheL1-environment, a user formulates some problem to be solve: for example, to prove a selected theorem. After this, the "evidence routine" begins to work. Since the information environment preserves a signature of an initial ForTheL-text and natural-type proof methods may be used by the "routine", the user has a possibility to control a search process.

At present, some units of such a "chain" of transformations already are constructed: a translator from ForTheL to ForTheL1 and a proving system for first-order logic are implemented. Units providing a proof representation, an interactive search, and a control of the information environment are under development now.

Let us give an example of a correct ForTheL-text, which deals with some elementary notions of the non-standard analysis taken from [4]. We omit definitions which are not

necessary for the proof of a given theorem. We just note that `nst[A]` represents a set A in a non-standard universe ${}^*\mathbf{U}$, and a predicate "`x is close to y`" affirms that a point `x` from ${}^*\mathbf{U}$ is close to a point `y` from the standard universe.

```
If set _ A is a subset of the set _ B
           then nst[A] is a subset of nst[B].
Definition 1. A is a subset of B iff all elements
              of the set _ A belong to the set _ B.
Definition 2. M is closed iff for all t if some element of
              nst [M] is close to t then t belongs to the set_M.
Definition 3. M is compact iff every element of nst[M]
              is close to some element of the set _ M.
Theorem 1. Closed subset of a compact set is compact.
```

(In ForTheL, the underscore symbol declares a variable following it.)

For the above ForTheL-text, the translator generates the Following ForTheL1-text. Note that there are new variables in it of the form *_i*). These variables are implicitly introduced by the quantifying words "`some`", "`every`", "`all`".

```
FORALL A (set[A] THEN FORALL B (set[B] THEN (subset[A, B]
         THEN subset[nst[A], nst[B]]))).
DEFINITION 1. FORALL A (set[A] THEN FORALL B (set[B]
              THEN (subset[A, B] IFF FORALL _1 (in[_1, A]
              THEN in[_1, B])))).
DEFINITION 2. FORALL M (set[M] THEN (closed[M] IFF
              FORALL t (EXISTS _2 (in[_2, nst[M]]
              AND close[_2, t]) THEN in[t, M]))).
DEFINITION 3. FORALL M (set[M] THEN (compact[M] IFF
              FORALL _3 (in[_3, nst[M]] THEN EXISTS _4
              (in[_4, M] AND close[_3, _4])))).
THEOREM 1. FORALL _6 ((set[_6] AND compact[_6]) THEN
           FORALL _5 ((subset[_5, _6] AND closed[_5])
           THEN compact[_5])).
```

3 A Deductive Technique

Deduction in EA-style requires a special sequent formalism using a proof environment constructed on the base of a ForTheL1-text containing definitions, auxiliary propositions, and a theorem to be proven. Here we note that the first paper on an EA-style heuristic procedure for theorem proof search in Group Theory appeared in 1966 [5]. In that paper an attempt was done to make allowance on a formal level for some proof search methods used in mathematical papers. Then that formal technique was extended to certain fragments of the Set Theory. Its final completion appeared as a specific calculus [6], which

was meant for ascertaining the validity of 1st-order classical logic formulas. Its further development gave arise to the first representative [7] of an EA- style family of a-sequent calculi, which later was "extended" to a number of a-sequent calculi (see, for example, [8, 9]). That is why a deductive technique of SAD is based on a subsequent modification of the formalism.

3.1 A Goal-Driven Calculus of E-Sequents

By now, an original prover using a computer-oriented modification of the a-sequent formalism has been constructed and has been implemented. The prover handles ForTheL1-sections, which can be treated as 1st-order logical formulas. It bases on a special calculus GD, which is the modification of a calculus gS from [8], has some changes in comparison with it, and reflects the main approach to the construction of the prover. Note that a basic object of GD is an e-sequent.

The most important changes in the modification concern quantifiers rules and handling the premises of e-sequents. In comparison with gS, the initial set of premises stays the same during the whole inference search. So, new premises cannot be added to an e-sequent in GD, though a set of equations is changed every time, when auxiliary goal rule applications are made. As to quantifier rules, we must note that to avoid the irrelevant duplications of premises, which are observed in gS, a special technique of bound variables processing is developed.

To do this modification self-contained enough and independent from [8], here we introduce all the necessary notions.

PRELIMINARIES. We consider classical first-order logic with the universal and existential quantifiers and with the propositional connectives of implication ($\supset$), disjunction ($\vee$), conjunction ($\wedge$), and negation ($\neg$). Atomic formulas are denoted below by A, B, C, or D, literals are denoted by L or M, formulas are denoted by F, G, P, or Q. All the letters can be subscripted.

The expression $F^{\neg}$ denotes the result of one-step carrying of the negation into a formula F.

We define *positive* ($P\lfloor F^{+}\rfloor$) *and negative* ($P\lfloor F^{-}\rfloor$) *occurences* of a formula F in a formula P in an usual manner.

Everywhere below W denotes a set of first-order formulas. We assume that no two quantifiers in formulas from W contain the same variable and treat free variables in formulas from W as constants. Thus, it may be considered, without loss of generality, that all variables in W are bound. Also, we assume that the notion of the scope of a quantifier is known to a reader.

A variable v is said to be *unknown w.r.t.* W if there exists a formula $P \in W$ such that $P\lfloor(\forall vF)^{+}\rfloor$ or $P\lfloor(\exists vF)^{-}\rfloor$ holds. Correspondingly, v is said to be *fixed w.r.t.* W if there exists $P \in W$ such that $P\lfloor(\forall vF)^{-}\rfloor$ or $P\lfloor(\exists vF)^{+}\rfloor$ holds. Obviously, all variables in formulas from W are either unknown or fixed (are "dummies" and "parameters" in the terminology of [10]).

A set W of formulas induces an antisymmetric relation $\prec_W$ on the bound variables from W by the following: $u \prec_W w$ holds if and only if, for some formula F from W, a quantifier from F containing w occurs in the scope of a quantifier from F containing u.

For every variable v from W, we introduce a countable set of new variables of the form ${}^k v$, where $k = 0, 1, 2, \ldots$ These new variables will be called *indexed*. For every expression U, an expression ${}^k U$ denotes the result of substitution of ${}^k v$ for every unindexed variable v in U.

An indexed variable ${}^k v$ is defined to be *unknown (fixed) w.r.t.* W if and only if the corresponding variable v is unknown (fixed) w.r.t. W. Further, we extend the relation $\prec_W$ to indexed variables and define that ${}^k u \prec_W {}^k w$ holds if and only if $u \prec_W w$.

We treat the notion of a substitution as in [11]. Any substitution component is considered to be of a form t/x, where x is a variable (denominator), and t is a term (numerator) of a substitution. Also, we assume that a reader is familiar with the notion of the common unifier of sets of expressions.

Let σ be a substitution and $t/u \in \sigma$, where u is an unknown variable w.r.t. W, and let the term t contain a variable w fixed w.r.t. W. In this case we define $w \ll_\sigma^W u$. Obviously, the relation $\ll_\sigma^W$ is antisymmetric.

A substitution s is said to be *admissible* for a set W of formulas if and only if (i) denominators of σ are unknown variables w.r.t. W, and (ii) the transitive closure of $\prec_W \cup \ll_\sigma^W$ is an antisymmetric relation.

Remark. It may be checked that the above admissible substitution notion is equivalent to the one from [12] (also, see [8]).

A *pasting substitution* for a set W of formulas is a substitution π, which satisfies the following condition: every its component has the form ${}^m v/{}^k v$, and for every variable $u \prec_W v$ it is true that ${}^m u/{}^k u \in \pi$.

An *equation* is a pair of terms s, t, written as $s \approx t$.

Assume L is a literal of the form $R(t_1, \ldots, t_n)$ $(\neg R(t_1, \ldots, t_n))$ and M is a literal of the form $R(s_1, \ldots, s_n)$ $(\neg R(s_1, \ldots, s_n))$, where R is a predicate symbol. Then $\Sigma(L, M)$ denotes the set of equations $\{t_1 \approx s_1, \ldots, t_n \approx s_n\}$. In this case L and M are said to be *equal modulo* $\Sigma(L, M)$.

THE CALCULUS GD. The basic object of a goal-driven calculus GD under consideration is an e-sequent, which may be considered as a special generalization of the standard notion of sequents and is closely connected with the notion of a-sequents from [8].

An *e-sequent* is an expression of the form $\Gamma \rightarrow \Delta, [\Lambda]\ \langle E \rangle$, where Γ and Δ are sequences of formulas, Λ is a sequence of literals, and E is a set of equations. The formulas from Γ are called *premises*, the formulas from Δ are called *goals*, and the literals from Λ are called *framed literals*. Here we consider e-sequents containing only one goal. We assume also that all variables occuring in Λ, Δ, and E are indexed, and all variables occurring in Γ are unindexed.

Axioms of the calculus GD are e-sequents of the form $\Gamma \rightarrow \#, [\Lambda]\ \langle E \rangle$.

Inference Rules. The calculus GD contains the following *inference rules*:

Goal-Splitting Rules (GS):

$$(\rightarrow \supset)_1\colon \quad \frac{\Gamma \rightarrow F \supset G, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow G, [\Lambda]\ \langle E\rangle} \qquad (\rightarrow \supset)_2\colon \quad \frac{\Gamma \rightarrow F \supset G, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow \neg F, [\Lambda]\ \langle E\rangle}$$

$$(\rightarrow \vee)_1\colon \quad \frac{\Gamma \rightarrow F \vee G, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow G, [\Lambda]\ \langle E\rangle} \qquad (\rightarrow \vee)_2\colon \quad \frac{\Gamma \rightarrow F \vee G, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow F, [\Lambda]\ \langle E\rangle}$$

$$(\rightarrow \wedge)\colon \quad \frac{\Gamma \rightarrow F \wedge G, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow F, [\Lambda]\ \langle E\rangle \qquad \Gamma \rightarrow G, [\Lambda]\ \langle E\rangle} \qquad (\rightarrow \neg)\colon \quad \frac{\Gamma \rightarrow \neg F, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow F^{\neg}, [\Lambda]\ \langle E\rangle}$$

$$(\rightarrow \forall)\colon \quad \frac{\Gamma \rightarrow \forall\, {}^{k}v\, F, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow F, [\Lambda]\ \langle E\rangle} \qquad (\rightarrow \exists)\colon \quad \frac{\Gamma \rightarrow \exists\, {}^{k}v\, F, [\Lambda]\ \langle E\rangle}{\Gamma \rightarrow F, [\Lambda]\ \langle E\rangle}$$

Auxiliary-Goal Rule (AG):

$$\frac{\Gamma_1, F\lfloor M^{+}\rfloor, \Gamma_2 \rightarrow L, [\Lambda]\ \ \langle E\rangle}{\Gamma_1, F, \Gamma_2 \rightarrow {}^{l}(\neg F), [L, \Lambda]\ \ \langle E \cup \Sigma(L, {}^{l}M)\rangle}$$

Termination Rule 1 (T1):

$$\frac{\Gamma_1, M, \Gamma_2 \rightarrow L, [\Lambda]\ \ \langle E\rangle}{\Gamma_1, M, \Gamma_2 \rightarrow \#, [\Lambda]\ \ \langle E \cup \Sigma(L, {}^{l}M)\rangle}$$

Termination Rule 2 (T2):

$$\frac{\Gamma \rightarrow L, [\Lambda_1, M, \Lambda_2]\ \ \langle E\rangle}{\Gamma \rightarrow \#, [\Lambda_1, M, \Lambda_2]\ \ \langle E \cup \Sigma(\widetilde{L}, M)\rangle}$$

In AG the formula F is not a literal. (So, T1 is necessary for completeness of GD.) In AG and T1 the literals L and ${}^{l}M$ are equal modulo $\Sigma(L, {}^{l}M)$, where the index l is a new index. In T2 the literals $\widetilde{L}$ and M are equal modulo $\Sigma(\widetilde{L}, M)$, where $\widetilde{L}$ denotes a literal L', if L is $\neg L\,'$, and $\widetilde{L}$ denotes $\neg L$ otherwise. L.)

Let $P_1, \ldots, P_n \rightarrow G$ be an usual sequent, where $P_1, \ldots, P_n, G$ are formulas that do not contain indexed variables. An *initial e-sequent*, induced by this sequent, is called the expession of the form $P_1, \ldots, P_n, \neg G \rightarrow {}^{0}G, [\ ]\ \langle\ \rangle$.

When a proof of an initial e-sequent S is searching, an *inference tree* Tr w.r.t. S is constructed. At the beginning of searching Tr contains only S. The subsequent nodes are generated by means of rules of GD. Inference tree grows "from top to bottom" in accordance with the order of inference rule applications.

An inference tree Tr is considered to be a *proof tree w.r.t.* S if and only if the following conditions are satisfied: (i) every leaf of Tr is an axiom, (ii) if E is a union of sets of equations from of all the leaves of Tr, then there exist a pasting substitution π and a substitution σ such that σ is a common unifier of all the equations from $E\pi$, where $E\pi$ denotes the result of the application of π to all the terms from E, and (iii) σ is admissible for the set of premises of S.

Whenever the AG rule is applied to some e-sequent with a literal goal L, we fix some positive occurrence of that literal (modulo equations) in some premise F and put the formula ${}^{l}(\neg F)$ in the goal of the derived e-sequent, preserving this fixed occurrence of L.

I.e. we require that any subsequent application of $(\rightarrow \supset)$ and $(\rightarrow \vee)$ preserves the fixed occurrence.

Proposition 1. *Let $P_1, \ldots, P_n$ be a consistent set of formulas and G be a formula. The sequent $P_1, \ldots, P_n \rightarrow G$ is deducible in Gentzen's calculus **LK** [13] if and only if there exists a proof tree w.r.t. the initiale-sequent $P_1, \ldots, P_n, \neg G \rightarrow {}^0G, [\ ]\ \langle\ \rangle$ in the calculus GD.*

Proof. A proof of this proposition may be obtained by extending a proof of the soundness and completeness of a calculus GD-2 [14] to the case of 1st order logic, taking into account peculiarities of admissible substitutions pointed in [12].

Proposition 2. *A formula G is valid if and only if there is exists a proof tree w.r.t. the initial e-sequent $\neg G \rightarrow {}^0G, [\ ]\ \langle\ \rangle$ in the calculus GD.*

Proof. It follows from the completeness of **LK** and Prop. 1.

3.2 Deduction in a ForTheL1-environment

We remind that after translation of a ForTheL-text to be processed into a correspondent ForThel1-text an assertion T to be proved is represented as a substantive ForTheL1-section "theorem", in which conditions Txt (including assumptions, definitions and auxiliary propositions) and a conclusion are separated, and an initial e-sequent (with respect to Txt and T) is constructed with Txt and T in its antecedent and succedent respectively. It was noted above that any ForTheL1-sentence can be treated as an analog of some 1st-order classical logic formula. It enables to construct formula patterns of such units of a ForTheL1-text as the theorem to be proved, a definition, and an auxiliary proposition and to treat a self-contained ForTheL1-text as a set of 1st-order formulas. So, it is possible to understand unambiguously such terms as "ForTheL1-text consistency", "logical consequence of a theorem from a given ForTheL1-text", and "validity" (of the theorem to be proved) without special defining the semantics of the ForTheL1-language. With this in mind, we state main results about GD as follows.

Corollary 3. *A ForTheL1-theorem T is a logical consequence of a consistent ForTheL1-text Txt (which does not include T) if and only if a proof tree (with an initial sequent w.r.t. T and Txt) can be constructed in the calculus GD.*

Corollary 4. *A ForTheL1-theorem T is valid if and only if a proof tree (with an initial sequent w.r.t. T only) can be constructed in the calculus GD.*

We note, as a side-result, that rather rich collection of rules in GD enables to construct various proof search strategies, which reflect proofs constructions from usual mathematical texts and allow a user to influence a proof process actively, when an interactive mode of proof search is used. If these strategies (with or without participation of a human) ensure an exhaustive search, then corollaries 1 and 2 guarantee the soundness and completeness of a strategy under consideration.

4 Related Work

There exists a number of projects and systems now which have intersections with the EA programme or which are close to it in ideas. Below, we give only a brief remark on such projects and systems.

The project MIZAR [15] is the most close to the EA programme. The objective of that project is the development of computer systems for mathematical texts processing. The collection of mathematical languages, which are convenient both for mathematicians and computer processing forms the basis of a particular MIZAR system. The foundation of a MIZAR mathematical language is the language of classical 1st-order logic. The correctness of a mathematical text can be checked by the system.

The main objective of the THEOREMA project [16] is to develop an integrate environment (both logical techniques and program tools) for solving mathematical problems (including numeric computations, algebraic transformations, and theorem proving). THEOREMA is designed to offer such an environment for a mathematician which enables to pass through the whole cycle of problem solving process.

The final objective of the QED project [17] is the development of an integrate distributed computer environment containing the totality of important and valid mathematical knowledge.

The system OMEGA [18] is developed in order to support theorem proving in mathematics and mathematical education. The system includes a proof planner and an integrated collection of tools for the forming of subproblems, searching for proofs for subproblems, and representing proofs. Both well-known general-purpose theorem provers and a computer algebra system are integrated into OMEGA as external units.

The system ISABELLE [19] is used as a tool for creating environment for interactive theorem proving. A mathematical knowledge base that includes the library of concrete mathematics and various packages for advanced mathematical concepts is exploited. The system supports the kind of proving usual for mathematicians by reasoning in the terms of a given application domain.

A number of the above and some other projects, systems, and groups (for example, the DReaM group, Mechanized Reasoning Group, CAAR group, etc.) are the members of the CALCULEMUS project [20] interested in the integration of the deductive and computational power of both deduction systems and computer algebra systems.

5 Conclusion

The linguistic and deductive peculiarities of SAD show that SAD has a specific technique of mathematical text processing, which takes into account expressive features of ForTheL as well as the "fashion" of logical inference search.

The language ForTheL can be used as a tool to write and verify mathematical papers and formal specifications, to perform model checking and so on. It also can be used both the universal interface for declarative mathematical knowledge bases and a tool for the integration of computer mathematical services.

As for the deductive technique of SAD, the calculus GD can serve as a good base for the further development of EA-style theorem proving in the direction of the efficiency improvement by means of formalizing some natural proof search methods such as, for example, definition and auxiliary proposition applications.

In this connection, the authors hope for that results obtained in the frame of investigations on SAD can be helpful in attacking the following problems : distributed automated theorem proving, checking self-contained mathematical texts for correctness, remote training in the mathematical disciplines, extracting knowledge from mathematical papers, and constructing knowledge bases for mathematical theories.

References

1. Glushkov, V., Kostyrko, V., Letichevski, A., Anufriyev, F., Asel'derov, Z. : On a language for description of formal theories (in Russian). In: Teoreticheskaya kibernetika **3** (1970).
2. Glushkov, V., Vershinin, K., et al. : On a formal language for description of mathematical texts (in Russian). In: Avtomatizatsiya poiska dokazatel'stv teorem v matematike. Institute of Cybernetics, Kiev (1974) 3–36.
3. Vershinin, K., Paskevich, A. : ForTheL — the language of formal theories. IJ Information Theories and Applications **7-3** (2000) 121–127.
4. Davis, M. : Applied non-standard analysis (Translated from English). Mir, Moskva (1980).
5. Anufriyev, F., Fediurko, V., Letichevski, A., Asel'derov, Z., Didukh, I. : On one algorithm of theorem proof search in Group Theory (in Russian). Kibernetika **1** (1966) 23–29.
6. Anufriyev, F. : An algorithm of theorem proof search in logical calculi (in Russian). In: Teoriya avtomatov **1**. Institute of Cybernetics, Kiev (1969).
7. Degtyarev, A., Lyaletski, A. : Logical inference in SAD (in Russian). In: Matematicheskiye osnovy sistem iskusstvennogo intellekta. Institute of Cybernetics, Kiev (1981).
8. Degtyarev, A., Lyaletski, A., Morokhovets, M. : Evidence Algorithm and Sequent Logical Inference Search. In: LNAI **1705** (1999) 44–61.
9. Degtyarev, A., Lyaletski, A., Morokhovets, M. : On the EA-style integrated processing of self-contained mathematical texts. Proc. of the Intern. Workshop CALCULEMUS'2000, Great Britain (2000).
10. Kanger, S. : Simplified proof method for elementary logic. In: Comp. Program. and Form. Sys.: Stud. in Logic. North-Holl., Publ. Co. (1963).
11. Robinson, J. : A machine-oriented logic based on resolution principle. In: J. of the ACM (1965) 23–41.
12. Lyaletski, A. : Gentzen calculi and admissible substitutions. In: Actes preliminaries, du Simposium Franco-Sovetique "Informatika-91". Grenoble, France (1991) 99–111.
13. Gentzen, G. : Untersuchungen uber das Logische Schliessen. Math. Zeit. **39** (1934) 176–210.
14. Lyaletski, A., Paskevich, A. : Goal-Driven Inference Search in Classical Propositional Logic. In: Proc. of the Inter. Workshop STRATEGIES'2001. Siena, Italy (2001).
15. http://mizar.org/
16. Buchberger, B., Jebelean, T., Kriftner, F., Marin, M., Tomuta, E., Vasaru, D. : A survey of the Theorema project. Proc. of ISSAC'97. Maui, Hawaii (1997) 384–391.
17. http://www.mcs.anl.gov/qed/
18. http://www.ags.uni-sb.de/~omega/
19. http://www.cl.cam.ac.uk/Research/HVG/Isabelle/
20. http://www.mathweb.org/calculemus/